#2 RS

190-1472

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN THE APPLICATION OF	)
	)
Duxbury	) Group Art Unit No.
	)
SERIAL NO: Not yet assigned	)
	)
FILED: Herewith	)
	)
FOR: Template Animation and Debugging Tool	)

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to "Commissioner of Patents, Washington, D.C. 20231", on March 8, 2001

Name of person signing Heather Vinson

Signature

CLAIM FOR PRIORITY

Honorable Commissioner of
Patents and Trademarks
Washington, D.C. 20231

Dear Sir:

Under the International Convention, for the purposes of priority, applicant claims the benefit of United Kingdom Application No. 0009013.4, filed April 13, 2000.

A certified copy of said application is appended hereto.

DATE: March 8, 2001

Respectfully submitted,

William M. Lee

William M. Lee, Jr.
Registration No. 26,935
LEE, MANN, SMITH, MCWILLIAMS
SWEENEY & OHLSON
P.O. Box 2786
Chicago, Illinois 60690-2786
(312) 368-1300
Fax (312) 368-0034

(USA)

PATENTS • DESIGNS • COPYRIGHT • TRADE MARKS
The Patent Office

INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed M Conkie

Dated 24 JAN 2001

Patents Form 1/77

Patents Act 1977
(Rule 16)

The Patent Office

1/77

THE PATENT OFFICE
B
13 APR 2000
NEWPORT

15APR00 E529280-1 D02057
P01/7700 0.00-0009013.4

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference C1457

2. Patent application number
(The Patent Office will fill in this part) **0009013.4**

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

INTERNATIONAL COMPUTERS LIMITED
26 Finsbury Square, London EC2A 1SL

Patents ADP number *(if you know it)* 28258008

If the applicant is a corporate body, give the country/state of its incorporation ENGLAND

4. Title of the invention

TEMPLATE ANIMATION AND DEBUGGING TOOL

5. Name of your agent *(if you have one)*

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

D C Guyatt
Intellectual Property Department
International Computers Limited
Stevenage
Herts
SG1 2DY

Patents ADP number *(if you know it)* 10493700 5

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

YES

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	
Description	11
Claim(s)	2
Abstract	1
Drawing(s)	2 + 2

10. If you are also filing any of the following, state how many against each item.

Priority documents	–
Translations of priority documents	–
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	–
Request for preliminary examination and search *(Patents Form 9/77)*	1
Request for substantive examination *(Patents Form 10/77)*	–
Any other documents *(please specify)*	Fee Sheet

11. I/We request the grant of a patent on the basis of this application.

Signature D C Guyatt Date 12/4/2000

12. Name and daytime telephone number of person to contact in the United Kingdom

D. C. Guyatt
01438 786235

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.

d) If you have answered 'Yes' Patents Form 7/77 will need to be filed.

e) Once you have filled in the form you must remember to sign and date it.

f) For details of the fee and ways to pay please contact the Patent Office.

Patents Form 7/77

Patents Act 1977
(Rule 15)

The Patent Office

7/77

THE PATENT OFFICE
K
23 JUN 2000
RECEIVED BY POST

Statement of inventorship and of right to grant of a patent

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference
C1457

2. Patent application number
(if you know it) GB 0009013.4

3. Full name of the or of each applicant

INTERNATIONAL COMPUTERS LIMITED

4. Title of the invention

TEMPLATE ANIMATION AND DEBUGGING TOOL

5. State how the applicant*(s)* derived the right from the inventor*(s)* to be granted a patent

BY REASON OF EMPLOYMENT

6. How many, if any, additional Patents Forms 7/77 are attached to this form?
(see note (c))

7. I/We believe that the person*(s)* named over the page *(and on any extra copies of this form)* is/are the inventor*(s)* of the invention which the above patent application relates to.

Signature D C Guyatt Date 22/6/00

8. Name and daytime telephone number of person to contact in the United Kingdom

D.C. Guyatt
01438 786235

Notes

a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there are more than three inventors, please write the names and addresses of the other inventors on the back of another Patents Form 7/77 and attach it to this form.

d) When an application does not declare any priority, or declares priority from an earlier UK application, you must provide enough copies of this form so that the Patent Office can send one to each inventor who is not an applicant.

e) Once you have filled in the form you must remember to sign and date it.

Enter the full names, addresses and postcodes of the inventors in the boxes and underline the surnames

Paul Duxbury
36 School Lane
Brereton
Sandbach
Cheshire
CW11 1RN

Patents ADP number *(if you know it)*: 6085369001

Patents ADP number *(if you know it)*:

Patents ADP number *(if you know it)*:

Reminder

Have you signed the form?

TEMPLATE ANIMATION AND DEBUGGING TOOL

Background to the Invention

This invention relates to a method and apparatus for animating and debugging electronic templates. The invention is particularly concerned with templates for generating text format documents, especially HTML (Hypertext Markup Language) documents.

As more and more websites start to contain functionality as well as static information, in some cases fronting major enterprise applications, it is becoming common to generate web pages independently for each user, dynamically on request. This is necessary so that results of queries or calculations can be inserted into the pages, and so that personalisation can be effected.

Traditionally this meant that building web pages became a development programming task. "CGI" programs were written to perform application functions and to output HTML responses. But this meant that programmers were also responsible for the look and feel of the site, normally the province of graphic designers. Also the simplest change to site design had to go back to the development programmers who created it.

To help this situation, some form of template-based rendering system is now often used. Here, a graphic designer generates HTML for the look and feel of a site, but leaves "holes" into which dynamic information can be placed. This is known as a *template*. When required, the template is rendered, by inserting the dynamic information into the holes, to generate pure HTML.

The main advantages of template-based rendering are:

- Separates programming and graphic design aspects, and hence the skills needed.

- Allows standard layouts, navigation, and house-styles to be easily imposed.
- Makes changes to look and feel easier, since only the templates need altering.
- Allows alternative renderings for different devices, by supplying multiple templates for the same content.
- In some cases, allows automatic construction of hyperlinks.

A template may consist of a document (typically in a text format such as HTML) containing embedded commands that identify what information is to be inserted into the template when it is rendered.

When such a template is being developed, it would be useful to be able to step through the template up to any specified command, and to display the partial result, i.e. the effect of rendering the template up to that command. This would provide a useful tool for debugging the template. For example, it could be used to "animate" the template, showing the effect of stepping through a sequence of commands. However, a problem with this is that the partial result at each step of the animation may not be well-formed; for example HTML end tags may be missing from the partial output. The object of the present invention is to provide a novel template animation tool which overcomes this problem.

Summary of the Invention

According to the invention a method for animating and debugging an electronic template containing embedded commands, comprises:

- rendering the template, up to a specified step number, to produce a partial output; and
- including in the partial output items subsequent to the specified step number, to ensure that the partial output is well-formed.

Brief Description of the Drawings

Figure 1 shows a computer system embodying the invention.

Figure 2 shows the logical organisation of a content store.

Figure 3 shows the user interface of a template animator tool.

Figure 4 shows a parse tree structure generated by the template animator tool.

Description of an Embodiment of the Invention

One embodiment of the invention will now be described by way of example with reference to the accompanying drawings.

Figure 1 shows a computer 10, which in this example is assumed to be used for developing new websites. A content store 11, holds the content for the websites. The computer can access the content store by way of a content store access service 12. The computer also includes a template renderer 13 and a template animation/debugging tool 14. The components 12-14 may be implemented as Java servlets.

Content store

The content store 11 holds all the content for the websites, including templates and partial results. It contains a set of objects, logically organised in a tree structure. Each object represents either an actual item of content (such as a template, dynamic information to be inserted into a template, or a rendered document), or a folder which may contain other objects. The content store may be distributed, and accessed over a network using the standard WebDAV (Web-based Distributed Authoring and Versioning) protocol, or alternatively may be local.

Each object in the content store has a hierarchic address, which identifies its position in the tree structure. For example, Figure 2 shows a portion of the content store, with objects identified by addresses such as "/sport/news/football". The root of the tree is indicated by "/".

Each object in the content store has an internal structure, comprising a content body, and a number of properties. The properties may be further organised into one or more property sheets, so that name clashes between standard properties and those assigned by different groups of individuals are avoided. Property sheets provide a convenient visualisation of the concept of XML namespaces as used in WebDAV.

The properties of an object can be addressed by appending a suffix of the form *:propertysheet:property* to the object address. For example,

/news/speeches/s1234:PUBLIC:speaker

addresses the *speaker* property on the PUBLIC property sheet of the object at /news/speeches/s1234. If the property sheet is not specified, the PUBLIC property sheet is assumed by default.

An object can model any of the following items:

- A simple file, where all the content is in the body, and is treated as just an unstructured row of bytes or text characters. There may be some fixed properties, such as content length and modification date, corresponding to those of an ordinary file.
- A document together with its metadata, i.e. information about the document such as its author, approval status, subject matter, default publishing template and so on.
- A fielded database record, where all the data is held in the properties, here having the role of database fields.
- Combinations of the above, e.g. a fielded database record with associated metadata.

Templates

A template consists of a document (typically HTML) containing embedded commands that identify what information is to be inserted into the template when it is rendered. These commands include WebDAV and other commands, embedded in the document using XML syntax. These embedded commands are distinguished from conventional HTML tags by a "ds:" namespace.

Templates may reside in file store, or may be held in the content store itself. Some examples of typical embedded commands that can be used in templates will now be described.

***insert* command**

The *insert* command retrieves or constructs some text, and then inserts it into an output stream. One possible format for the *insert* command is:

<ds:insert content="*SourceAddress*" />

The *SourceAddress* attribute specifies the content store address of an object or property whose contents are to be retrieved and inserted into the output stream.

For example, the command:

<ds:insert content="/sport/news/000216" />

retrieves the news article at address "/sport/news/000216" from the content store, and inserts it into the output stream.

Content properties can also be directly addressed, using the suffix notation mentioned above. For example:

<ds:insert content="/sport/news/000216:headline" />

inserts the headline property associated with the news article.

The *content* attribute may be replaced by a *src* (source) attribute. This indicates a URL (Universal Resource Locator) which can be used to access an object from an external website.

for command

The *for* command is used to specify an iterative loop through a set of objects or values, repeating some processing for each. One possible format for this command is:

```
<ds:for content="RootObject" filter="Filter" >
     Loop Body
</ds:for>
```

This command causes the enclosed *Loop Body* text to be repeated a number of times, once for each object in the *RootObject* folder. The *Filter* attribute is an expression involving comparison operators, which specifies a condition for selecting objects from this set. For example, the filter expression:

```
subject EQ football OR subject EQ golf
```

selects objects whose *subject* property is equal to either "football" or "golf".

For example, the construction:

```
<ds:for content="/sport/news" filter="this:subject EQ
'football'">
     ...
</ds:for>
```

loops through all the articles in folder /sport/news, selecting only those whose *subject* property is equal to "football". This may be used, for example, to build an index page of news items relating to football.

A number of other "programming" commands (loops, conditions, procedures, variables etc.) are also provided, which may be used to produce very sophisticated and adaptive web pages.

Template renderer

A call to the template renderer 13 specifies the address in the content store of the template to be rendered. When called, the template renderer accesses the content store to get the specified template. It then parses the template, to identify any embedded commands in it. Each command is executed, and any text generated by the command is appended to an output string. Any parts of the template that are not embedded commands are simply copied to the output string. The resulting output string is passed back to the caller.

Template animation tool

The template animation tool 14 is web-based, using a conventional browser such as Microsoft Internet Explorer Version 4 to provide its user interface. As shown schematically in Figure 3, the user interface has four windows: control window 31, template window 32, view window 33 and watch window 34.

The control window 31 allows an object/template combination to be specified, or alternatively these may be preset according to the context when the animator is entered, e.g. by a button on a particular web page or administration screen. A toolbar with "VCR" type controls is also provided, for allowing the user to move to a particular step number within the template. These controls include buttons to allow single step (i.e. move forward to the next command), or "fast forward" (i.e. multiple steps, say five at a time). It is also possible to jump to the beginning or end. The resulting step number is displayed in a box. Alternatively, a step number can be entered explicitly in the box.

The template window 32 displays a portion of the template currently being rendered. All the embedded commands are highlighted, and the command which is about to execute is

highlighted in a different colour, font or style, or any other suitable method of distinction. Clicking on any of the embedded commands in this window causes a pop-up help window to be displayed for that command.

The view window 33 displays the result of rendering the template, up to the selected step number. As will be described, rather than simply truncating the template at the current command, the underlying HTML is adjusted to make it well-formed, i.e. to include end tags. This avoids most problems which might arise from passing incomplete HTML sequences to the browser. Through the control window, an option is available to view the rendered HTML text instead of the resulting page image.

The output view contains small hyperlinks known as "locators"; in this embodiment they are represented by small coloured circles containing the letter "L", although it will be appreciated that any other suitable form may be used. There is one such locator for each embedded command instance in the template, and each indicates the location of the output that was generated by the corresponding command instance. The locator's tool tip shows the command instance and the step number. Clicking on a locator will automatically "rewind" the rendering process and step to the selected command. This is useful for quickly finding the command which generated a particular (possibly erroneous) item of output.

Because it is not possible to put a link within another link, the locators for any commands occurring within links have to be queued until the link in which it is embedded closes. The same is true for marking template expressions that occur within the attributes of HTML tags, and doubly true if the tag happens to be a link.

The watch window 34 displays the current values of selected local variables. A default set of variable names can be

configured, and others can be entered into the Control Window. This feature is useful for debugging more complex scripts.

As described above, the animation tool enables a particular step number to be specified, by way of the control window, or through the locators. When a step number has been specified in either of these ways, the template renderer is called to render the template, starting from the beginning of the template, up to the specified step number. It should be noted that the rendering always restarts from the beginning of the template, even if the "step backwards" or "fast back" button is selected, although the operation will normally be so fast that the user will perceive it as stepping backwards.

The template rendering process will now be described in more detail.

First, the template renderer parses the template, including both HTML and embedded commands, and constructs a parse tree, which branches at each tag in the template that has a corresponding end tag: i.e. HTML tags such as <table> and <tr>, and also embedded commands such as the <ds:for ... > command described above. For example, consider the following simple HTML file:

```
<html>
  <head>
  <title>Sample Template</title>
  </head>
  <body>
    <table>
      <tr>
        <td><ds:insert content="A" /></td>
        <td><ds:insert content="B" /></td>
      </tr>
      <tr>
        <td><ds:insert content="C" /></td>
```

```
            <td><ds:insert content="D" /></td>
          </tr>
        </table>
        ...
      </body>
    </html>
```

It can be seen that this HTML will generate a 2 X 2 table. The data items in the table are retrieved from the content store locations A to D and inserted into the output document as specified by the <ds:insert ... /> commands.

Figure 4 shows the parse tree generated as a result of parsing this file. It can be seen that this tree does not contain any HTML end tags; rather, they are implied from the tree structure.

Next, the template renderer expands the embedded commands, e.g. by replacing the <ds:insert> commands with the appropriate data items from the content store. If the embedded commands contain a loop (for example, using the <ds:for> command described above), the loop is iterated as required.

The renderer keeps track of the number of expansion steps it has performed, and stops when the specified step number is reached, even if this is part way through a loop. (Each iteration around a loop counts as a step). When the required step number is reached, the renderer truncates the remainder of the tree by removing everything later than the last command to be expanded, i.e. everything to the right of the path between the root and this command. For example, referring to Figure 4, if the last command to be expanded was the one shown in bold, then everything to the right of the path shown in bold lines in will be truncated.

The template renderer then serialises the truncated parse tree, so as produce expanded HTML representing the partial result of

rendering the template up to the specified step. In this example, the expanded HTML will be as follows:

```
<html>
  <head>
  <title>Sample Template</title>
  </head>
  <body>
    <table>
      <tr>
        <td>Data A</td>
        <td>Data B</td>
      </tr>
    </table>
  </body>
</html>
```

It can be seen that this expanded HTML has all the required end tags (such as </table>) and hence is well-formed HTML.

Finally, the expanded HTML is executed and the partial result is displayed in the view window. In this example, only the first row of the table will be displayed. Alternatively, if the option to view the rendered HTML text was selected in the control window, the expanded HTML itself is displayed in the view window.

It will be appreciated that many modifications may be made to the system described above without departing from the scope of the present invention as defined by the claims.

CLAIMS

1. A method for animating and debugging an electronic template containing embedded commands, comprising:

- rendering the template, up to a specified step number, to produce a partial output ; and
- including in the partial output items subsequent to the specified step number, to ensure that the partial output is well-formed.

2. A method according to Claim 1 wherein the items subsequent to the specified step number include end tags.

3. A method according to Claim 2, including the steps:

- parsing the template to generate a parse tree having a branch at each tag for which there is a corresponding end tag;
- truncating the parse tree to remove parts subsequent to the specified step number; and
- forming the partial output from the truncated parse tree.

4. A method according to any preceding claim wherein the template is an HTML document.

5. A method according to any preceding claim including inserting locator markers in a display of the partial output, each indicating the location of material that was generated from a particular command.

6. A method according to Claim 5 wherein selecting one of the locator markers automatically rewinds the display to the command corresponding to that marker.

7. A method according to Claim 5 or 6 wherein any locator marker corresponding to a command embedded within a link is queued until the link in which it is embedded closes.

8. A method for animating and debugging an electronic template substantially as hereinbefore described with reference to the accompanying drawings.

9. A computer system substantially as hereinbefore described with reference to the accompanying drawings.

ABSTRACT

An animation tool is used for debugging electronic templates (e.g. for a Web page) containing embedded commands. The tool renders the template, up to a specified step number, to produce a partial output. The partial output includes items subsequent to the specified step number (e.g. HTML end tags), to ensure that the partial output is well-formed. The tool parses the template to generate a parse tree having a branch at each tag for which there is a corresponding end tag. It then truncates the parse tree to remove parts subsequent to the specified step number, and forms the partial output from the truncated parse tree.

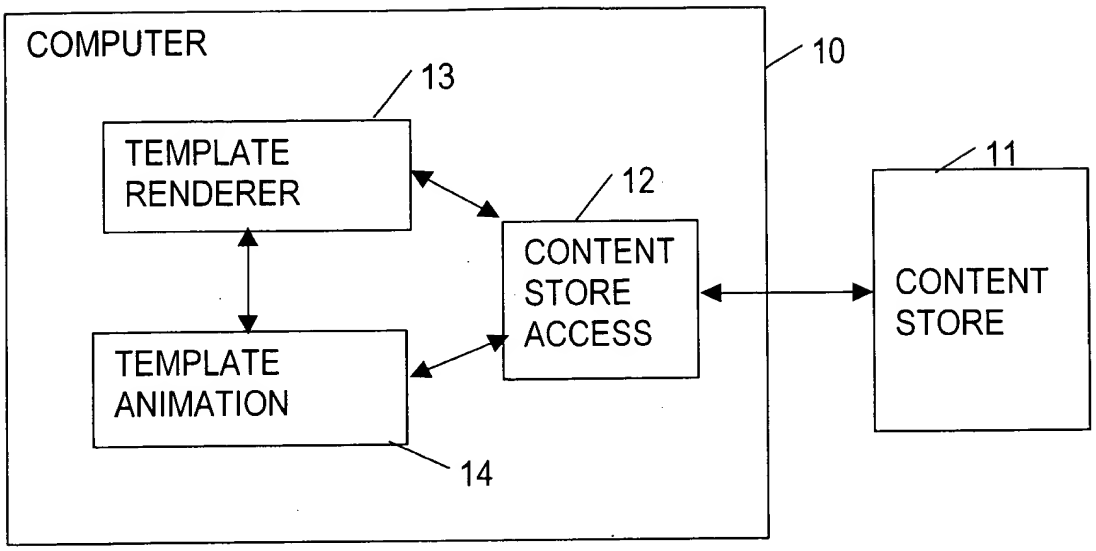
COMPUTER
10
13
TEMPLATE RENDERER
12
CONTENT STORE ACCESS
11
CONTENT STORE
TEMPLATE ANIMATION
14

FIG. 1

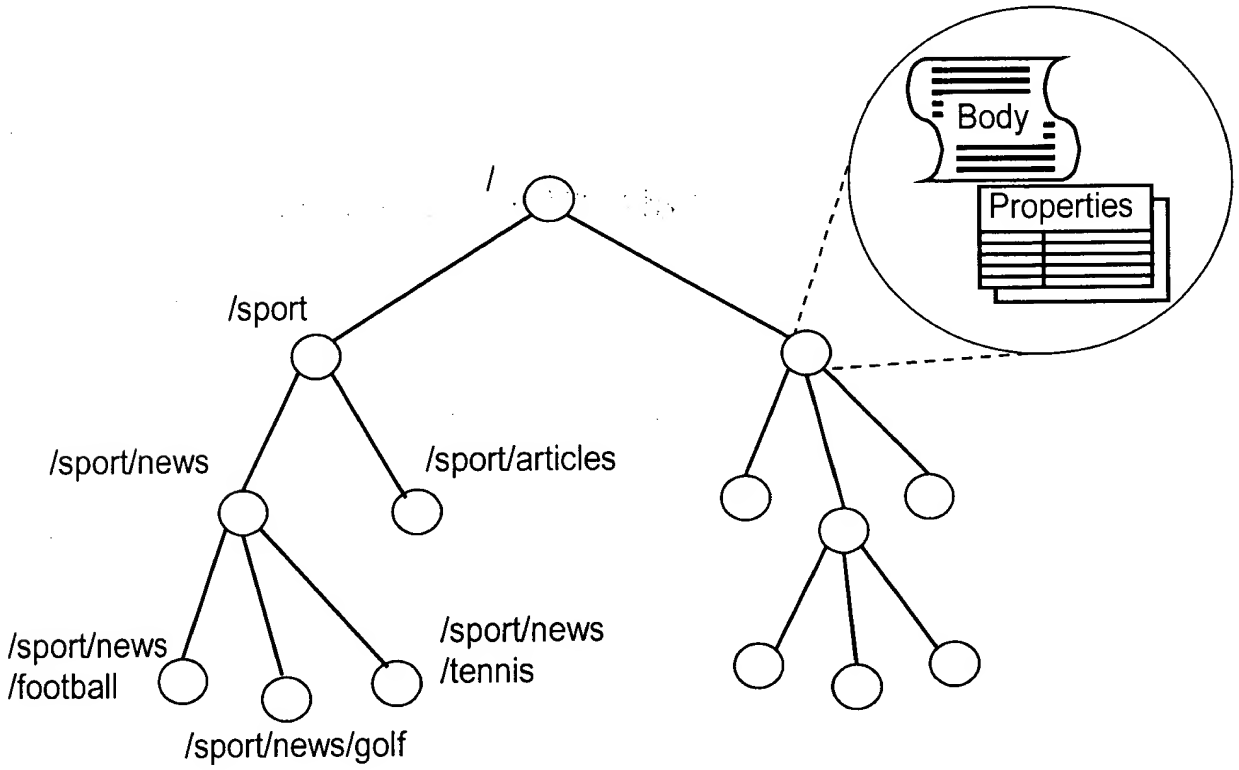
Body
Properties
/
/sport
/sport/news
/sport/articles
/sport/news /football
/sport/news /tennis
/sport/news/golf

FIG. 2

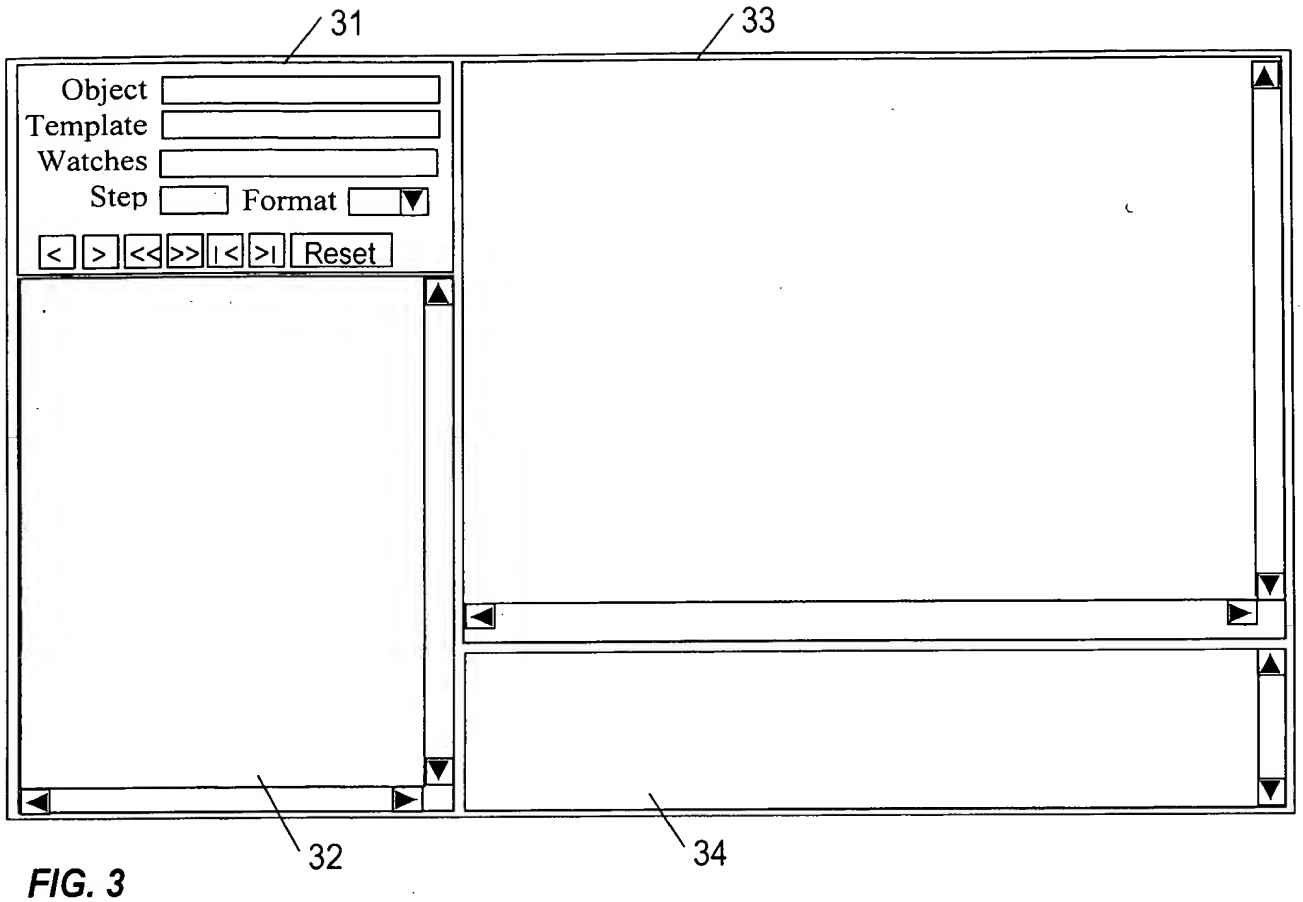
31
33
Object
Template
Watches
Step
Format
< > << >> |< >| Reset
32
34

FIG. 3

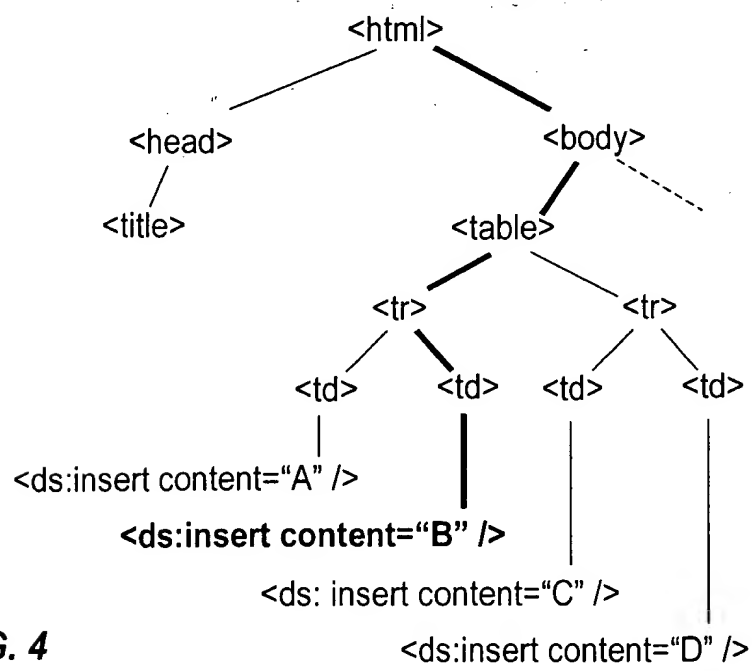
<html>
<head>
<body>
<title>
<table>
<tr>
<tr>
<td>
<td>
<td>
<td>
<ds:insert content="A" />
<ds:insert content="B" />
<ds: insert content="C" />
<ds:insert content="D" />

FIG. 4